AFCRL- 66-781

A MATCHING PROCEDURE FOR W-ORDER LOGIC

by

William Eben Gould

APPLIED LOGIC CORPORATION
ONE PALMER SQUARE
PRINCETON, NEW JERSEY

Contract No. AF 19(628)-3250

Project No 8672

Scientific Report No. 4

October 15, 1966

This research was sponsored by the Advanced Research Projects Agency under ARPA Order No. 700-

Prepared for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS 01730

AFCRL- 66-781

A MATCHING PROCEDURE FOR W-ORDER LOGIC

by

William Eben Gould

APPLIED LOGIC CORPORATION
ONE PALMER SQUARE
PRINCETON, NEW JERSEY

Contract No. AF 19(628)-3250

Project No 8672

Scientific Report No. 4

October 15, 1966

This research was sponsored by the Advanced Research Projects Agency under ARPA Order No. 700

Prepared
for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS 01730

PREFACE

This paper was presented as a doctoral dissertation to Princeton University, under the supervision of Dr. James R. Guard of Princeton University and Applied Logic Corporation. Portions of the work were financed by a Science Faculty Fellowship from the National Science Foundation while the author was on leave from Washington College.
The problem investigated is pertinent to the project on "Semi-Automated Mathematics" being carried out by Applied Logic, originally for the Air Force, Cambridge Research Laboratories and currently for the Advanced Research Projects Agency. More specifically, the problem is proposed in the report listed as [3] in the bibliography.

TABLE OF CONTENTS

ABSTRACT

Formation rules are presented for an ω-order logic with λ operator in which each term has a well defined type. Transformation rules are given defining λ-conversion, special conversion, substitution, and alphabetic change of bound variables. The notion of one term being an "instance" of another is then defined using these transformation rules. The main problem of the paper is to develop a "matching procedure" for pairs of terms, so that every common instance of the two given terms is an instance of some term produced by the matching procedure. This is accomplished by analysing the terms according to the formation rules, matching the outermost parts first, then proceeding inductively inward. Examples are given to show that an infinite set of terms may be needed for this purpose. A procedure is developed for producing at least one common instance, if one exists.

CHAPTER 1

ω -ORDER LOGIC

Section 1. Formation Rules

The basic system with which we shall be dealing is ω-order logic, wherein each term has a well-defined type. The formation rules for types are quite simple. We shall have (at least) two basic types, P and Q, which are intended to be the types of propositions and individuals, respectively. If f is a function of n variables with types $T_1, \ldots T_n$ respectively, and with values of type T_{n+1}, then f has type $(T_1, \ldots T_{n+1})$. More formally, our rules will be:

(1) P is a type,

(2) Q is a type,

(3) if $T_1, \ldots T_{n+1}$ are types (for n=1, 2, ...) then $(T_1, \ldots T_{n+1})$ is a type.

The formal system shall contain the following atomic symbols:

(1) infinitely many variables and constants of every type;

(2) punctuation symbols -), (,] , [, and , ;

(3) quantifiers - A, E, and λ (Church's λ operator, see [1]).

Among the constants of type (P, P, P) we may have the usual propositional connectives, $\supset$, $\vee$, &, and $\equiv$, which may be employed in the usual fashion (between arguments), and among the constants of type (P, P) we may have $\sim$, negation.

We shall present rules for the construction of well formed terms (WFTs), each of which has a type. A WFT of type P will also be called "a well-formed formula (WFF)".

(1) Any constant or variable of type T is a WFT of type T.

(2) If $W_1, \ldots W_n$ are WFTs of types $T_1, \ldots T_n$ respectively and V is a WFT of type $(T_1, \ldots T_{n+1})$, then $[V](W_1, \ldots W_n)$ is a WFT of type T_{n+1}. W_1 will be called "the i-th argument of V".

(3) If W is a WFF and x is a variable, then $(Ax)(W)$ and $(Ex)(W)$ are WFFs.

(4) If $x_1, \ldots x_n$ are variables of types $T_1, \ldots T_n$ respectively and W is a WFT of type T_{n+1}, then $(\lambda x_1, \ldots x_n)(W)$ is a WFT of type $(T_1, \ldots T_{n+1})$.

We shall adopt the following conventions on metavariables:

b, c, d, e represent types;

f, g, ... z represent (system) variables, with the informal agreement f, g, and h will be preferred when the functional nature of the variable is to be stressed;

B, C, D, F, ...N represent constants (A and E being reserved for quantifiers);

P, Q, R, ... Z represent WFTs.

Any metavariable may occur with or without subscripts or ', and any variable occurring as a subscript is a numerical variable. Variables representing terms may occur with component well formed parts (WFPs) displayed in parentheses, with substitution indicated as usual by replacement of displayed symbols. Note that $P(x_1, \ldots x_n)$ is the same WFT as P, the notation simply calling attention to the variables $x_1, \ldots x_n$ which may occur inside P. $[P](x_1, \ldots x_n)$, however, represents the result of applying the function P to the arguments $x_1, \ldots x_n$. Parentheses or brackets may be omitted when there is no possibility of confusion.

Whenever an algorithm calls for the introduction of a new variable, we assume it to be the first unused variable of the proper type in alphabetic order.

Section 2. Conversion Rules

We need not be concerned with rules of inference in general, but there are four conversion rules which we must define.

Alphabetic change of bound variable: If y does not occur in P, then from $(Ax)P$ we may get $(Ay)P'$, where P' is the result of replacing every free occurrence of x in P by y. The analogous rules apply to $(Ex)P$ and $(\lambda x_1, \ldots x_n)P$.

Substitution: If R and T are WFTs and x is a variable of the same type as T, then we may replace every free occurrence of x in R by T, provided no free variable of T is captured by a quantifier of R. The operation of replacing x by T will be written "$x \rightarrow T$". The usual provision can be made for simultaneous substitution.

λ conversion: From $[(\lambda x_1, \ldots x_n)R](T_1, \ldots T_n)$, we may get R' where R' is derived from R by $x_1 \rightarrow T_1, \ldots x_n \rightarrow T_n$, unless some free variable of T_1 would be captured by this substitution. In that case, we first apply alphabetic change of bound variable to the appropriate well formed part of R. Note that x_1 and T_1 must have the same type in order for the original expression to be well formed.

Special conversion: From $(\lambda x_1, \ldots x_n)([R](x_1, \ldots x_n))$, we may get simply R, and vice versa, provided no x_1 occurs free in R.

If a WFT, B, is derivable from a WFT, C, by successive applications of substitution to all of C and of alphabetic change of bound variable, λ conversion, and special conversion to WFPs of C, then B will be

called "an instance of C". If B is an instance of D and also an instance of C, then B will be called "a common instance of D and C", and if D and C have any common instance, they will be said to "match".

Section 3. Matching

Our primary concern in this paper will be to find a method of generating common instances for pairs of WFTs. There are several reasons why this may be important. Suppose we wish to draw a conclusion from premises R and $S \supset T$. Suppose further that S' is a common instance of R and S, and that T' is a corresponding instance of T (results from applying the same substitutions to T as were applied to S in deriving S'). Since any instance of a WFF is a logical consequence of that WFF, we may conclude T'. This sort of procedure, called matching, is one of the basic techniques in the semi-automated mathematics of [2], [3], and [4].

Other possible applications might involve substitutivity of equality. Suppose we have two assertations, R = T and W, and some WFP, R', of W matches R. We may then apply the matching substitutions to W and replace the resulting instance of R by the corresponding instance of T.

It has been shown in [3] that in first order logic any two formulas P and Q which match have a general matching formula, R, with the properties:

(1) R is a common instance of P and Q

(2) every common instance of P and Q is an instance of R.

Thus, in an obvious sense, R is the most general common instance of P and Q.

An analogous concept for higher order logic is that of a general matching set (GMS). If P and Q match, then a set S of WFTs is said to be a GMS for P and Q if:

(1) Every WFT in S is a common instance of P and Q.

(2) Every common instance of P and Q is an instance of some WFT in S.

(3) No WFT in S is an instance of any other WFT in S.

Unfortunately, it is not true that every pair of WFTs which match have a GMS. Therefore, we shall introduce the concept of a major matching set (MMS) for P and Q, which satisfies conditions (1) and (2) above, but not necessarily (3). Obviously (except to intuitionists), every pair of matching WFTs does have a MMS; namely, the set of all common instances. It is also obvious that any pair of WFTs which have a finite MMS have a GMS. Simply run through the MMS in any order, discarding any WFTs which are instances of other WFTs in the set.

However, it does not follow that any infinite MMS can be reduced to a GMS. Indeed, we shall exhibit several pairs of WFTs which have infinite MMSs and no GMSs at all, finite or infinite.

CHAPTER 2

THE MATCHING PROCEDURE

Section 1. Preliminaries

Our objective in this chapter will be to develop an algorithm, called the matching procedure, which will produce a MMS for any given pair of WFTs. The MMS produced may be infinite, or it may be empty, in which case the WFTs do not match. Chapter 3 will attack (with partial success) the problem of predicting when an infinite MMS is necessary. Chapter 4 will handle the problem of discovering, in finitely many steps, whether the MMS is empty or not.

The matching procedure will be described inductively, according to the formation rules on page 2. Note, however, that in matching two WFTs we may need to know more than just the MMSs for their component parts. We may need the substitutions which lead to those MMSs. Two essentially different substitutions may lead to the same matching WFT, but if the variables replaced occur elsewhere in our overall WFTs, the distinction between these substitutions must be maintained (see example 1). Naturally any substitutions called for must be legitimate in the larger context. For example, B(x) and B(y) can be matched by $x \to y$, but (Ey)B(x) and (Ey)B(y) do not match.

Throughout our matching procedure any variable which is replaced in one WFT must be replaced in the same fashion in the other WFT. This is to prevent undoing the matching work which has already been carried out. Since no such restriction is present in the definition of "common instance", we shall start the matching procedure by replacing each free variable of our first WFT which also occurs in our second WFT by an unused variable of the

proper type. Since this substitution is reversible, we have not changed the set of all instances of the WFT. For some applications it may be desirable to suppress this operation and insist on having the same variable occur free in both WFTs.

In order to keep our WFTs in as standard a form as possible, we shall employ λ conversion whenever it can be applied. Also, we shall employ special conversion whenever it shortens the WFT.

The following techniques will be useful.

Direct elimination: Suppose S is a WFP of T and S occurs in the i-th argument of a n-place variable f. Make the substitution $f \rightarrow (\lambda u_1, \ldots u_n)\, g(u_1, \ldots u_{i-1}, u_{i+1}, \ldots u_n)$. The resulting WFT will closely resemble T except for the absence of an occurrence of S (and of any other WFT which occurred elsewhere as the i-th argument of f). This procedure is said to "(directly) eliminate S from T".

There is also an indirect method of eliminating S from T, which causes more drastic changes in T. It will be discussed at the end of the chapter.

Direct reduction: Suppose P is $f(W_1, \ldots W_n)$ and P has the same type as W_i. The substitution $f \rightarrow (\lambda u_1, \ldots u_n)u_i$ is said to "(directly) reduce P to W_i". Not only will P be replaced by W_i as a result of this substitution, but any other occurrence of f with arguments will be replaced by its i-th argument.

Indirect reduction: Suppose P is $f(W_1, \ldots W_n)$, P and W_i have type b, and W_j has type $(c_1, \ldots c_p, b, c_{p+2} \ldots c_m, b)$. Suppose further that the identity function $(\lambda u_1, \ldots u_m)u_{p+1}$ matches (hence is an instance of) W_j. Then the substitutions $f \rightarrow (\lambda u_1, \ldots u_n)u_j(f_1(u_1, \ldots u_n), \ldots f_p(u_1, \ldots u_n), u_1, f_{p+2}(u_1, \ldots u_n), \ldots f_m(u_1, \ldots u_n))$ and $W_j \rightarrow (\lambda u_1, \ldots u_m)u_{p+1}$ are said

to "(indirectly) reduce P to W_i ". P will be replaced by W_i as a result of these substitutions, but occurrences of f with other arguments may retain vestiges of those arguments.

Indeed, if some other argument of f, W_k, also matches an identity function of the proper type, W_i can be buried deeper in the expression which replaces f. Assume for simplicity that W_i and $W_k (j \neq k)$ have the type of singulary functions. Then the substitutions $f \rightarrow (\lambda u_1, \ldots u_n) u_k(u_j(u_1)))$, $W_k \rightarrow (\lambda u)u$, and $W_j \rightarrow (\lambda v)v$ will reduce $f(W_1, \ldots W_n)$ to W_i. Obviously it may be possible to "bury" W_j even deeper than this, but this 'burying' process is limited by n. Because of type considerations, no occurrence of a u_m may be utilized within the scope of another occurrence of u_m. In such a case, we could replace the intervening functions by identity functions, leaving u_m (or W_m) as an argument of itself, which is impossible.

See Example 1 for an example of direct and indirect reduction.

The purpose of reduction is to transform P into some WFP, Q, of P, without introducing Q or any part of Q via a substitution. This will be particularly useful if Q contains a variable which is bound by some quantifier whose scope includes all of P.

We shall now proceed by cases, according to the forms of the WFTs to be matched. Every WFT either:

(1) is a single variable or constant;

(2) begins with a quantifier, A, E, or λ; or

(3) begins with a functional variable or constant followed by arguments.

Since the formation and transformation rules are identical for the quantifiers A and E, we may safely ignore A and treat only E. Note that in the WFT [P] $(T_1, \ldots T_n)$, P cannot begin with E. Furthermore, we may assume P does not begin with λ, since we could then apply λ conversion.

In general, we may safely treat P as if it were atomic. That is, we shall consider $f(T_1, \ldots T_n)$ but not $[[\ g(T_1, \ldots T_i)\ ] \ldots]\ (T_j, \ldots T_n)$. The general case will always follow in the same manner as the special case. Note that these two WFTs are instances of each other, using the substitutions:

$$f \rightarrow (\lambda u_1, \ldots u_n)([[\ g(u_1, \ldots u_i)\ ] \ldots]\ (u_j, \ldots u_n))$$

$$g \rightarrow (\lambda u_1, \ldots u_i) \ldots (\lambda u_j, \ldots u_n) f(u_1, \ldots u_n).$$

Likewise we shall, with one exception, discuss $F(T_1, \ldots T_n)$ but not $[[\ G(T_1, \ldots T_i)\ ] \ldots]\ (T_j, \ldots T_n)$. In Effect, we shall treat all the T's in the last WFT as arguments of G.

Whenever we assert that some substitution $f \rightarrow T$ is to be made, it is on the condition that f and T have the same type.

Section 2. Atomic WFTs

First, let us match a single constant, B, to a WFT, P. Clearly, in order to have a match, P must be:

(1) B,

(2) x, or

(3. $f(T_1, \ldots T_n)$.

In case (2), substitute $x \rightarrow B$. In case (3), either substitute $f \rightarrow (\lambda u_1, \ldots u_n)B$, or reduce P to some WFP of the form (1), (2), or (3).

Example 1. Match B to f(x, y), where B and y have type b, x has type (b, b), and f has type ((b, b), b, b). There are three ways to perform the match.

(1) $f \rightarrow (\lambda u, v)B$

(2) $f \rightarrow (\lambda u, v)v$, $y \rightarrow B$; using direct reduction

(3) $f \rightarrow (\lambda u, v)u(v)$, $x \rightarrow (\lambda w)w$, $y \rightarrow B$; using indirect reduction.

The WFT produced is the same, B, in each case. However, we would need all three substitutions if the given WFTs were imbedded in, say, C(f, b) and C(f, f(x, y)), since none of the three WFTs which replace f is an instance of any of the others.

Next, let us match a single free variable, x, to a WFT, P. If x does not occur in P, simply substitute $x \rightarrow P$. There is nothing to be gained by transforming P into x, as if x were constant. Any such transformation can still be applied after the WFTs have been matched by $x \rightarrow P$, giving exactly the same result. See example 3. However, transforming P into x and then substituting $x \rightarrow P$ may not give the same result as simply substituting $x \rightarrow P$. If x does occur in P, we must either eliminate it and proceed as above, or reduce P to x. The direct elimination of x is described above, the indirect elimination is postponed until the last section of this chapter.

If x is a bound variable which must be matched to some WFT, Q, then we may neither substitute for x, nor introduce x via substitution. Consequently we must either eliminate x altogether or else reduce Q, directly or indirectly, to some x which is already a WFP of Q. Similar remarks apply to the matching of any bound variable with arguments, say $x(S_1, \ldots S_n)$, to some WFT, Q. (See section 4.)

Section 3. WFTs which begin with a Quantifier

To match $(\exists x)P(x)$ and $(Ey)Q(y)$, we first apply alphabetic change of bound variable to get $(Ez)P(z)$ and $(Ez)Q(z)$. Then match P and Q (remembering, of course, that no substitution is allowed for z). The only other WFTs which can match $(Ex)P$ are y and $f(T_1, \ldots T_n)$. The former has been covered above and the latter will be covered in Section 4.

In similar fashion, we start the match of $(\lambda x_1, \ldots x_n)P$ to $(\lambda y_1, \ldots y_n)Q$ by applying alphabetic change of bound variable. In addition, we can match $(\lambda x_1, \ldots x_n)P$ to R by applying special conversion to R to get $(\lambda x_1, \ldots x_n)([R](x_1, \ldots x_n))$, provided R is of the proper type. (If x_i occurs in R we change to some new variable z_i in both WFTs). Then we must match P and $[R](x_1, \ldots x_n)$. Also, we may match $(\lambda x_1, \ldots x_n)P$ to x as above and to $f(T_1, \ldots T_m)$ as in Section 4.

Section 4. Functional Constants with Arguments

Next let us match $B(P_1, \ldots P_n)$ to $B(Q_1, \ldots Q_n)$. To do so we first match P_1 and Q_1, making the same substitutions in the remaining arguments. Then match the new second arguments, P_2' and Q_2', and so forth. For ease of application, it may be preferable in some examples to match the arguments in a different order.

Example 2. Match $K(x, B(x))$ and $D(C(y, D), B(C(E, z,))$. First match x to $C(y, D)$. This can only be done by $x \rightarrow C(y, D)$. Next we must match $B(C(y, D))$ to $B(C(E, z))$. This requires $y \rightarrow E$ and $z \rightarrow D$. The only common instance then is $K(C(E, D), B(C(E, D)))$.

Obviously $B(P_1, \ldots P_n)$ will not match $C(Q_1, \ldots Q_m)$.

Next, let us consider $f(T_1, \ldots T_n)$ and $C(S_1, \ldots S_m)$. The general procedure is rather complicated, so we will start with an example.

Example 3. Match P, which is $f(x, B)$, to Q, which is $C(y)$, where x, y, $C(y)$, and B all have the same type. There are two separate approaches. First we try to reduce P to some FP which matches Q. This can only

be done by $f \rightarrow (\lambda u, v)u$. To complete the match we must substitute $x \rightarrow C(y)$. The other approach in effect replaces f by C composed with some arbitrary new function. Specifically, substitute $f \rightarrow (\lambda u, v)C(g(u, v))$. P is thus replaced by $C(g(x, B))$, and we must match $g(x, B)$ to y. This can be done by any of the following:

(1) $y \rightarrow g(x, B)$

(2) $g \rightarrow (\lambda u, v)y$

(3) $g \rightarrow (\lambda u, v)u; \quad y \rightarrow x$ (or $x \rightarrow y$)

(4) $g \rightarrow (\lambda u, v)v; \quad y \rightarrow B$

Substitutions (2), (3), and (4) may all be discarded as special cases of (1). (Follow (1) by $g \rightarrow (\lambda u, v)y, g \rightarrow (\lambda u, v)u$, or $g \rightarrow (\lambda u, v)v$ respectively.) To summarize, we have two ways of matching P and Q:

(1) $f \rightarrow (\lambda u, v)u; \quad x \rightarrow C(y); \quad P, Q \rightarrow C(y)$

(2) $f \rightarrow (\lambda u, v)C(g(u, v)); \quad y \rightarrow g(x, B); \quad P, Q \rightarrow C(g(x, B))$

Here (1) and (2) are independent, since neither $(\lambda u, v)u$ nor $(\lambda u, v)C(g(u, v))$ is an instance of the other.

Now let us return to the general case of matching P, which is $f(T_1, \ldots T_n)$, to Q, which is $C(S_1, \ldots S_m)$. It is clear that either C must be introduced via substitution for f, or else it must be introduced into some T_i or found to be already there. (If instead of a constant, C, we whould have a bound variable, x, then only the last of these is possible.) The most general possible substitution in the former case is $f \rightarrow (\lambda u_1, \ldots u_n)C(f_1(u_1, \ldots u_n), \ldots f_m(u_1, \ldots u_n))$. In this way $f(T_1, \ldots T_n)$ is replaced by C composed with m functions of $T_1, \ldots T_n$. We then have to match in succession $f_i(T_1, \ldots T_n)$ and S_i, for $i = 1, 2, \ldots m$.

In the latter case, P must have the same type as T_i and T_i must match some initial segment of Q (with brackets removed, unless the initial segment is all of Q). We shall now resort to a (partial) analysis of the

hierarchy of the "arguments" of C. Suppose Q is $[C(S_1, \ldots S_p)]$ $(S_{p+1}, \ldots S_m)$ and T_i matches $C(S_1, \ldots S_p)$. Then we must substitute $f \rightarrow (\lambda u_1, \ldots u_n)(u_i(f_{p+1}(u_1, \ldots u_n), \ldots f_m(u_1, \ldots u_n)))$. P is thus replaced by $T_i(f_{p+1}(T_1, \ldots T_n), \ldots f_m(T_1, \ldots T_n))$. To complete the match we must match in succession T_i and $C(S_1, \ldots S_m)$, $f_{p+1}(T_1, \ldots T_n)$ and S_{p+1}, $f_m(T_1, \ldots T_n)$ and S_m ...

Example 4. Match f(x,y) and [B(C)] (D), where f has type (d,(d,e),e), x has type d, y has type (d,e), B has type(c,(d,e)), C has type c, and D has type d. No pertinent relations hold among types b, c, d, and e. First we try $f \rightarrow (\lambda u, v)$ [B(g(u,v))] (h(u,v)), replacing each "argument" of B in [B(C)] (D) by an arbitrary function of two variables. We must follow up by matching g(x,y) to C and h(x,y) to D. The first can only be accomplished by $g \rightarrow (\lambda u, v)C$. There are two ways to accomplish the second: $h \rightarrow (\lambda u, v)D$ and $h \rightarrow (\lambda u, v)u$, $x \rightarrow D$.

Since the second argument of f matches an initial WFP (minus the brackets, of course) of [B(C)] (D), we must also try $f \rightarrow (\lambda u, v)(v(h(u, v)))$. Then we must match y to B(C) and h(x,y) to D. The first obviously requires $y \rightarrow B(C)$, and the second leads to the same two substitutions as before. We have then a total of four ways to effect the match:

(1) $f \rightarrow (\lambda u, v)$ [B(C)] (D)

(2) $f \rightarrow (\lambda u, v)$ [B(C)] (u); $x \rightarrow D$

(3) $f \rightarrow (\lambda u, v)$ [v] (D); $y \rightarrow B(C)$

(4) $f \rightarrow (\lambda u, v)$ [v] (u); $y \rightarrow B(C)$; $x \rightarrow D$

The four sets of substitutions are independent, but in each case the resulting WFT is [B(C)] (D).

With minor modifications the same procedure as above can be applied to match P, which if $f(T_1, \ldots T_n)$, to Q, which is $(\lambda u_1, \ldots u_m)S$, or to R, which is (Ex)S. In this case no proper initial segment of Q or R

can be well formed, so the situation is a little simpler. If some T_j matches Q or R, we must try reducing P to that T_j. Also, we must try $f \rightarrow (\lambda v_1, \ldots v_n)(\lambda u_1, \ldots u_m)g(v_1, \ldots v_n)$ or $f \rightarrow (\lambda v_1, \ldots v_n)(\lambda x)g(v_1, \ldots v_n)$, respectively. With either of these substitutions, we must complete the match by matching $g(T_1, \ldots T_n)$ and S.

Section 5. Distinct Functional Variables with Arguments

Next, let us match $f(S_1, \ldots S_m)$ and $g(T_1, \ldots T_n)$. Again we shall give an example first.

Example 5. Match P, which is f(K, B, C(D)) to Q, which is g(K(B), C, D). First we must match each argument of f to an arbitrary function of the arguments of g, and vice versa. There are six matches to consider:

(1) K to g'(K(B), C, D). The only match comes from $g' \rightarrow (\lambda u, v, w)K$.

(2) B to g'(K(B), C, D). Substitute $g' \rightarrow (\lambda u, v, w)B$.

(3) C(D) to g'(K(B), C, D). The matching substitutions are:

$g' \rightarrow (\lambda u, v, w)C(D)$

" v(D)

" v(w)

" C(w).

(4) K(B) to f'(K, B, C(D)). The matching substitutions are:

$f' \rightarrow (\lambda u, v, w)K(B)$

" u(B)

" u(v)

" K(v)

(5) C to f'(K, B, C(D)). Substitute $f' \rightarrow (\lambda u, v, w)C$.

(6) D to f'(K, B, C(D)). Substitute $f' \rightarrow (\lambda u, v, w)D$.

Now we put all these matching substitutions together, using the appropriate bound variables instead of the actual arguments of f and g . We get:

$f \rightarrow (\lambda u, v, w)h(u, v, w, w, w, v, K(B), u(B), u(v), K(v), C, D);$

$g \rightarrow (\lambda u, v, w)h(K, B, C(D), v(D), v(w), C(w), u, u, u, u, v, w);$

$P, Q \rightarrow h(K, B, C(D), C(D), C(D), C(D), K(B), K(B), K(B), K(B), C, D)$.

Any other matching substitution will be a special case of this one.

Incidently, this example disproves the conjecture that $f(S_1, \ldots S_m)$ and $g(T_1, \ldots T_n)$ can be matched by regarding f and g alternately as held constant. If g is held constant here, we can not use the the third argument of f . That is, we must use a substitution such as $f \rightarrow (\lambda u, v, w)g(u(v), C, D)$. Likewise, if f is held constant, we can not use the first argument of g .

Returning to the general problem of matching P , which is $f(S_1, \ldots S_m)$, to Q , which is $g(T_1, \ldots T_n)$, here too we must match each S_k to $g_k(T_1, \ldots T_n)$ and each T_j to $f_j(S_1, \ldots S_m)$. With regard to types, if each S_k has type b_k and each T_j has type c_j, then g_k must be a variable of type $(c_1, \ldots c_n, b_k)$ and f_j must be a variable of type $(b_1, \ldots b_m, c_j)$. Unfortunately, we may not be able to combine all these matches into one pair of substitutions for f and g , since they may involve different substitutions for variables inside the arguments of f and g .

Suppose $x_1, \ldots x_d$ are all the free variables of the S's and T's . Let $P_{j,1}(S_1', \ldots S_m'), \ldots P_{j,a_j}(S_1', \ldots S_m')$ be any set of WFTs from a MMS for $f_j(S_1, \ldots S_m)$ and T_j, where S_p' is the instance of S_p which results from the substitutions used in deriving the WFT in which it lies. Likewise, let $Q_{k,1}(T_1', \ldots T_n'), \ldots Q_{k,b_k}(T_1', \ldots T_n')$ be any set of WFTs from a MMS for $g_k(T_1, \ldots T_n)$ and S_k . Now we must reconcile the different substitutions made for $x_1, \ldots x_d$ in all these WFTs. Suppose $R_1, \ldots R_c$ are all the substitutions made for x_1 in the various matches. We must find a

MMS for this entire set of WFTs , extending our matching procedure inductively from pairs of WFTs to c-tuples of WFTs . Let $W_1, W_2, \ldots$ be such a (possibly infinite) MMS . Then we adopt the substitution $x_1 \rightarrow W_q$ and proceed to reconcile the substitutions for x_2, then for x_3, etc. For each q , etc., we have a matching substitution in:

$$f \rightarrow (\lambda u_1, \ldots u_m) h(P_{1,1}(u_1, \ldots u_m), \ldots P_{1,a_1}(u_1, \ldots u_m), \ldots P_{n,a_n}(u_1, \ldots u_m) u_1, u_1, \ldots u_m);$$

$$g \rightarrow (\lambda u_1, \ldots u_n) h(u_1, \ldots u_1, \ldots u_n, Q_{1,1}(u_1, \ldots u_n), Q_{1,2}(u_1, \ldots u_n), \ldots Q_{m,b_m}(u_1, \ldots u_n));$$

$$x_1 \rightarrow W \; ; \quad x_2 \rightarrow \text{etc.}$$

We return now to our MMSs for S_k and $g_k(T_1, \ldots T_n)$ and for T_j and $f_j(S_1, \ldots S_m)$, choosing a new combination of WFTs from them. It is not actually necessary to try every combination. If all the substitutions for x's required by one match are also required by another, there is nothing to be gained by trying a combination of matches which includes the second but not the first.

Before we give another example, we will show that every match between $f(S_1, \ldots S_m)$ and $g(T_1, \ldots T_n)$ can be effected by substitutions of the form:

$$f \rightarrow (\lambda u_1, \ldots u_m) h(P_1(u_1, \ldots u_m), \ldots P_a(u_1, \ldots u_m), u_1, \ldots u_m);$$
$$g \rightarrow (\lambda v_1, \ldots v_n) h(v_1, \ldots v_n, Q_1(v_1, \ldots v_n), \ldots Q_b(v_1, \ldots v_n)).$$

Where, of course, $P_1(T_1, \ldots T_n)$ is to be matched to S_1, etc. All such substitutions are special cases of those described in the preceding paragraph, since those were constructed from MMSs for $g_1(T_1, \ldots T_n)$ and S_1, etc.

Let $P'(S_1', \ldots S_m')$ which is the same as $Q'(T_1', \ldots T_n')$, be a common instance of P , $f(S_1, \ldots S_m)$, and Q , $g(T_1, \ldots T_n)$, where S_1' is the instance of S_1 which results from the substitutions used in deriving P'

from P, etc. Consider the way in which the S_k' and T_j' are nested in P'. Each occurrence of S_k' is either (1) a proper part of an occurrence of a T_j', or (2) an expression $Q_a(T_1', \ldots T_n')$ in zero or more T_j's. Let us number the occurrences of S_k' in (2) as ${}_1S_1'$, ${}_2S_1'$, $\ldots$ ${}_cS_m'$ so that ${}_aS_k'$ is $Q_a(T_1', \ldots T_n')$. Any occurrence of a T_j' which is not part of such an expression must be an expression $P_b(S_1', \ldots S_m')$ in zero or more S_k's. Number these occurrences as above so that ${}_bT_j'$ is $P_b(S_1', \ldots S_m')$.

Now P' can be regarded as $Z({}_1S_1', {}_2S_1', \ldots {}_cS_m', P_1(S_1', \ldots S_m'), \ldots P_d(S_1', \ldots S_m'))$, which is also $Z(Q_1(T_1', \ldots T_n'), Q_2(T_1', \ldots T_n'), \ldots Q_c(T_1', \ldots T_n'), {}_1T_1', \ldots {}_dT_n')$. We can match P and Q by the substitutions:

$$f \to (\lambda u_1, \ldots u_m)h(u_1, u_1, \ldots u_m, P_1(u_1, \ldots u_m) \ldots P_d(u_1, \ldots u_m));$$

$$g \to (\lambda u_1, \ldots u_n)h(Q_1(u_1, \ldots u_n), Q_2(u_1, \ldots u_n), \ldots Q_c(u_1, \ldots u_n), u_1, \ldots u_n);$$

$$S_k \to S_k'; \quad T_j \to T_j'.$$

The original matching substitution is an instance of this match gotten through $h \to (\lambda u_1, \ldots u_{c+d})Z(u_1, u_2, \ldots u_{c+d})$.

Example 6. Match P, f(x), and Q, g(B,y), where x and B have type b and y has type c. We must use the following:

(1) Match $f_1(x)$ and B by $f_1 \to (\lambda u)B$.

(2) " " " $f_1 \to (\lambda u)u, x \to B$.

(3) Match $f_2(x)$ and y by $y \to f_2(x)$.

(4) Match $g_1(B, y)$ and x by $x \to g_1(B, y)$.

Since (1) does not require any substitutions for x or y, we may use it in each substitution for f and g. Using (1) alone, we have:

$$f \to (\lambda u)h(B); \quad g \to (\lambda u, v)h(u); \quad P, Q \to h(B).$$

Combining (1) and (2), we have:

$f \rightarrow (\lambda u)h(B, u)$; $g \rightarrow (\lambda u, v)h(u, u)$; $x \rightarrow B$; P, Q $\rightarrow h(B\ B)$.

Combining (1) and (3), we have:

$f \rightarrow (\lambda u)h(B, f_2(u))$; $g \rightarrow (\lambda u, v)h(u, v)$; $y \rightarrow f_2(x)$; P, Q $\rightarrow h(B, f_2(x))$.

Combining (1), (2), and (3), we have:

$f \rightarrow (\lambda u)h(B, u, f_2(u))$; $g \rightarrow (\lambda u, v)h(u, u, v)$; $x \rightarrow B$; $y \rightarrow f_2(x)$,

i.e., $y \rightarrow f_2(B)$; P, Q $\rightarrow h(B, B, f_2(B))$.

Combining (1) and (4), we have:

$f \rightarrow (\lambda u)h(B, u)$; $g \rightarrow (\lambda u, v)h(u, g_1(u, v))$; $x \rightarrow g_1(B, y)$; P, Q $\rightarrow h(B, g_1(B, y))$.

To combine (1), (2), and (4), we must reconcile the different substitutions for x in (2) and (4) by matching B and $g_1(B, y)$. This may be done by $g_1 \rightarrow (\lambda u, v)B$ or by $g_1 \rightarrow (\lambda u, v)u$, but in either case we end up substituting $x \rightarrow B$, $f \rightarrow (\lambda u)h(B, u, u)$, and $g \rightarrow (\lambda u, v)h(u, u, u)$, which is clearly redundant.

In combining (1), (3), and (4), we encounter a subtler conflict of substitutions. After substituting $y \rightarrow f_2(x)$, $g_1(B, y)$ becomes $g_1(B, f_2(x))$. We can not simply substitute $x \rightarrow g_1(B, f_2(x))$, but must match x and $g_1(B, f_2(x))$. Due to a difference of types, the latter can not be reduced to x , but x can be eliminated by $g_1 \rightarrow (\lambda u, v)g_2(u)$ or by $f_2 \rightarrow (\lambda u)z$. The first elimination leads to the match:

$f \rightarrow (\lambda u)h(B, u, f_2(u))$; $g \rightarrow (\lambda u, v)h(u, g_2(u), v)$;

$x \rightarrow g_2(B)$; $y \rightarrow f_2(g_2(B))$; P, Q $\rightarrow h(B, g_2(B)$,

$f_2(g_2(B)))$.

This is more general than the combination of (1), (2), and (3), as can be seen by making g_2 the identity function. The second elimination leads to the match:

$f \rightarrow (\lambda u)h(B, u, z)$; $g \rightarrow (\lambda u, v)h(u, g_1(u, v), v)$;

$x \rightarrow g_1(B, z)$; $y \rightarrow z$; P, Q $\rightarrow h(B, g_1(B, z), z)$.

It is not really necessary to use a new variable, z, in this case. The match can also be accomplished by:

$f \rightarrow (\lambda u)h(B, u, y)$; $g \rightarrow (\lambda u, v)h(u, g_1(u, v), v)$;

$x \rightarrow g_1(B, y)$; P, Q $\rightarrow h(B, g_1(B, y), y)$.

Section 6. The Same Functional Variable with Different Arguments.

Finally let us match $f(S_1, \ldots S_n)$ and $f(T_1, \ldots T_n)$. This is one case which may lead to an infinite MMS. We start with an example.

Example 7. Match P, which is $B(f(x,M),f,x,y)$, and Q, which is $B(f(y,N),f,x,y)$, where x and y are singulary function variables of the same type. To match $f(x,M)$ and $f(y,N)$ we must either eliminate the second argument of f directly, or "hide" it inside the first argument, and then replace the first argument. There are two ways of eliminating M and N directly:

$f \to (\lambda u,v)h$; P, Q $\to$ R, R being $B(h,(\lambda u,v)h,x,y)$;

and

$f \to (\lambda u,v)h(u)$; $x \to y$; P, Q $\to$ W, W being

$B(h(y),(\lambda u,v)h(u),y,y)$.

For every n, n=1,2,3,..., we can also match P and Q by:

$f \to (\lambda u,v)g_n(u,u(f_1(u,v)),\ldots u(f_n(u,v)))$;

$x \to (\lambda u)r$; $y \to (\lambda u)r$; P, Q $\to Y_n$, Y_n being

$B(g_n((\lambda u)r,r,\ldots r),(\lambda u,v)g_n(u,u(f_1(u,v)),$

$\ldots u(f_n(u,v))),(\lambda u)r,(\lambda u)r)$.

For $k<j$, Y_k is an instance of Y_j but Y_j is <u>not</u> an instance of Y_k. Using the fact that $\{R, W, Y_1, Y_2, \ldots\}$ is a MMS for P and Q, we shall show that P and Q have no GMS. Assume $\{Z_1, \ldots Z_n, \ldots\}$ is a GMS for P and Q. Some Z_b must be an instance of some Y_c, since R and W alone obviously do n for a MMS. Y_{c+1} is in turn an instance of some Z_d. If d=b, then Y_{c+1} is an instance of Y_c. This is false. If d=b, Z_b is an instance of Z_d, by transitivity. This contradicts the definition of a GMS. Hence P and Q have <u>no</u> GMS.

For the general case of matching P, $f(S_1 \ldots S_n)$, and Q, $f(T_1, \ldots T_n)$, we shall assume that S_j and T_j match for $1 \leq j \leq m$, but do not match for $m < j \leq n$. We must consider every substitution for f of the form

$$f \rightarrow (\lambda u_1, \ldots u_n) g(u_1, \ldots u_m, u_1(f_{1,1,1}(u_1, \ldots u_n), f_{1,1,2}(u_1, \ldots u_n), \ldots f_{1,1,p_1}(u_1, \ldots u_n)), u_1(f_{1,2,1}(u_1, \ldots u_n), \ldots f_{1,2,p_1}(u_1, \ldots u_n)), \ldots u_2(f_{2,1,1}(u_1, \ldots u_n), \ldots f_{2,1,p_2}(u_1, \ldots u_n)), \ldots u_m(f_{m,1,1}(u_1, \ldots u_n), \ldots)).$$

The first subscript on $f_{h,j,k}$ is the same as that of the u_h in whose scope it occurs, the second is free to run from 0 (vacuous) to ∞, and the bounds on the third are determined by the type of u_h. Some of the u_h's may be missing. The match is completed by matching simultaneously the corresponding arguments of g.

To show that this gives a MMS, assume R is a common instance of P and Q. Then R must have been obtained from P and Q by substituting $f \rightarrow (\lambda u_1, \ldots u_n) V(u_1, \ldots u_n)$. We may assume that u_k, $m < k \leq n$, occurs only within the scope of some u_j, $1 \leq j \leq m$. Any other occurrence of u_k must be eliminated by a substitution for a free variable of V (since S_k and T_k do not match), and such a substitution can be made at the beginning to get a new V. Therefore V is an expression in $u_1, \ldots u_m$, with and without arguments, the arguments being expressions in $u_1, \ldots u_n$. V is thus an instance of an expression of the type given in the above paragraph.

Section 7. Another Use of Indirect Elimination

This technique of indirectly eliminating an argument by hiding it inside another argument can also be applied to the problem of matching a single variable to a WFT. Details of the general case are omitted, as they are quite similar to what we have just been through.

Example 8. Match $B(x, f)$ and $B(f(x, y), f)$. We must either reduce $f(x, y)$ to x by $f \rightarrow (\lambda u, v)u$, or else eliminate x by:

$$f \rightarrow (\lambda\, u, v)h(v, v(f_1(u)), \ldots\ v(f_n(u))),$$

$$y \rightarrow (\lambda\, u)z$$

$$x \rightarrow h((\lambda\, u)z, z, z, \ldots\ z).$$

Once again we have an infinite MMS, n being an arbitrary integer.

CHAPTER 3

CONDITIONS FOR THE MMS TO BE INFINITE

Section 1. Necessary Conditions

Let us examine the conditions under which the matching of $f(S_1, \ldots S_n)$ and $f(T_1, \ldots T_n)$ leads to an infinite MMS. We assume that these WFTs occur as parts of larger WFTs, P and Q. Otherwise a GMS results from the single substitution $f \rightarrow (\lambda u_1, \ldots u_n)h$. Also, we assume that the context of a WFP under discussion does not require its elimination in the matching process. For example, we will not consider the occurrences of f in the corresponding WFTs g(A(f)) and g(B(f)).

As shown by Example 7, the key step in establishing an infinite MMS is "hiding" some argument inside another. Accordingly, the following three conditions are necessary for our MMS to be infinite.

First, there must be something to hide. One possibility is that some arguments, say, S_n and T_n, do not match, so that their elimination, direct or indirect, is essential to any match. The other possibility is that S_n and T_n match, but that matching them requires an otherwise unnecessary substitution for some variable which also occurs outside S_n and T_n. If neither of these is the case, we may match all the S_j and T_j in succession, as if f were a constant.

As an alternative, we may have to eliminate some variable because it occurs in the wrong place, as in Example 8. If it is necessary to eliminate S_n from $f(S_1, \ldots S_n)$ because it contains a variable, x, which is to be matched to all of $f(S_1, \ldots S_n)$, we may treat the case as if x were $f(S_1, \ldots S_{n-1}, T)$, where T does not match S_n. Indeed, after eliminating S_n, we shall replace x by whatever remains of $f(S_1, \ldots S_n)$.

Second, there must be a hiding place. Some arguments, say S_1 and T_1, must match in a way consistent with the matching of the rest of P and Q. Furthermore, for some k, it must be possible to eliminate u_k from $[S_1](u_1, \ldots u_p)$ and $[T_1](u_1, \ldots u_p)$. Otherwise it would be necessary to eliminate the non-matching arguments, S_n and T_n, directly.

Third, there must be something left of the hidden arguments after the match is completed. This means that f must occur somewhere in P or Q without S_1 or T_1 as its first argument. Since the arguments to be hidden in S_1 and T_1 are to be completely eliminated eventually, they could just as well be eliminated from $f(S_1, \ldots S_n)$ and $f(T_1, \ldots T_n)$ directly if f did not also occur with some other first argument or with no arguments at all.

Section 2. Sufficient Conditions

If we assume in addition that all other occurrences of f in P and Q are without arguments and do not need to be matched to other WFTs, then we have sufficient conditions for the MMS of P and Q to be infinite, provided P and Q match at all. For simplicity of notation, assume now and hereafter S_1 and T_1 have the type of singulary functions.

Let R_q be $(\lambda u_1, \ldots u_n)h_q(u_1, u_1(f_1(u_1, \ldots u_n)), \ldots u_1(f_q(u_1, \ldots u_n)))$. We can match P and Q by substituting $f \rightarrow R_q$, followed by the substitutions which eliminate the arguments of S_1 and T_1 and match the rest of P and Q.

This match is not an instance of any match derived from $f \rightarrow (\lambda u_1, \ldots u_n)$ $h_s(u_1, \ldots u_m, u_1(f_1(u_1, \ldots u_n)), \ldots u_1(f_s(u_1, \ldots u_n)), u_2(g_1(u_1, \ldots u_n)), \ldots)$, where $s < q$. If u_n is not an argument of h_s in the latter WFT (and it can not be if S_n and T_n do not match), then there is no way to introduce $u_1(f_{s+1}$ $(u_1, \ldots u_n)), \ldots u_1(f_q(u_1, \ldots u_n))$ as arguments in passing from h_s to h_q. If u_n is an argument of h_s, then S_n and T_n must be matched, which requires

a substitution somewhere else in P and Q which was not necessary following $f \rightarrow R_q$. Thus we need to make substitutions of arbitrarily great length for f in order to get a MMS for P and Q. As in Example 7, there will be no GMS.

The above argument is not affected if, besides occurring without arguments and with arguments $S_1, \ldots S_n$ and $T_1, \ldots T_n$, f occurs also in the WFTs $f(X_1, \ldots X_n)$ and $f(Y_1, \ldots Y_n)$, which are to be matched to each other, with the following provisos. X_1 and Y_1 must match (consistantly with the match of P and Q), and either for some k it is possible to eliminate u_k from $[X_1]\ (u_1, \ldots u_p)$ and from $[Y_1]\ (u_1, \ldots u_p)$, or else X_n and Y_n match.

If f occurs only with arguments, the situation is quite complicated. Some examples follow.

Example 9. Match $B(f(x, M), f(z, w), x, y)$ and $B(f(y, N), f(z, w), x, y)$. We shall list the matching substitutions and the resulting WFTs.

$f \rightarrow (\lambda\, u, v)h$

R: $B(h, h, x, y)$

$f \rightarrow (\lambda\, u, v)\, g_0(u); \quad x \rightarrow y$

Y_0: $B(g_0(y), g_0(z), y, y)$

$f \rightarrow (\lambda\, u, v) g_n(u, u(f_1(u, v)), \ldots u(f_n(u, v)));$

$x \rightarrow (\lambda\, u)r; \quad y \rightarrow (\lambda\, u)r$

Y_n: $B(g_n((\lambda\, u)r, r, \ldots r), g_n(z, z(f_1(z, w), \ldots z(f_n(z, w))), (\lambda\, u)r, (\lambda\, u)r)$

Here $\{R, Y_0, Y_1, \ldots\}$ is a MMS, but in this case Y_{n+1} is an instance of Y_n gotten by substituting $g_n \rightarrow (\lambda\, u_1, \ldots u_n) g_{n+1}(u_1, u_1(f_1(u_1, w), \ldots u_1(f_n(u_1, q)))$. For n=0, add also the substitution $y \rightarrow (\lambda\, r)r$. Thus $\{R, Y_0\}$ is a finite GMS for these two WFTs.

Example 10. Match the following:

$(Ew)B(f(x, M), f(z, w), x, y)$

$(Ew)B(f(y, N), f(z, w), x, y)$

We get a MMS here by using exactly the same substitutions as in Example 9, but in this variation we can not introduce w via a substitution in order to reduce Y_{n+1} to Y_n. Therefore there is no finite GMS.

Example 11. Match the following:

$B(f(x, M), f(z, C), f(w, D), x, y)$

$B(f(y, N), f(z, C), f(w, D), x, y)$

The technique of Example 9 which showed Y_{n+1} to be an instance of Y_n will not work here, because D and C are different constants. Again there is no finite GMS.

CHAPTER 4

FINDING A SINGLE COMMON INSTANCE

Section 1. A Method which will not Work

Although an infinite set of substitutions may be required to find a MMS when we have to eliminate an argument from $f(S_1, \ldots S_n)$, it is to be hoped that a finite set will suffice to tell whether or not our WFTs match at all. For instance, it seems reasonable that we could find a match, if any existed, by using each eligible argument as a "hiding place" only once. That is, we should try the substitution $f \rightarrow (\lambda u_1, \ldots u_n)h(u_1, \ldots u_m, u_1(f_1(u_1, \ldots u_n)), u_2 f_2(u_1, \ldots u_n)), \ldots u_m(f_m(u_1, \ldots u_n)))$.

However, it turns out that there is no way of telling how many of our possible substitutions we must try before we find the first match, unless we examine the complete WFTs to be matched. The following example demonstrates this.

Example 12. The WFTs to be matched are f(x,B) and f(x, C), as imbedded in:

P_n: $(\lambda y, z)D(f(x, B), f(y, z))$

Q_n: $(\lambda y, z)D(f(x, C), G(y, y(H_1(z)), \ldots y(H_n(z))))$

Since y and z are bound variables, they may not be introduced by substitution. In order to match the second arguments of D, therefore, we must tuck the second argument of f inside the first argument n times. The shortest matching substitution which we can use for the first arguments of D, f(x, B) and f(x, C), is therefore $F \rightarrow (\lambda u, v)g(u, u(h_1(v)), \ldots u(h_n(v)))$. We can then complete the match by substituting $g \rightarrow G, h_1 \rightarrow H_1, \ldots h_n \rightarrow H_n$. Since n is an arbitrary integer, we have established our claim.

The use of bound variables in this example can be avoided by a change similar to that used in passing from Example 10 to Example 11.

We can even modify Example 12 to show that examining all the WFPs of the WFTs to be matched which contain f or are to be matched to WFPs containing f, will not be sufficient to tell how long a substitution we need for f. Let P and Q be as follows:

P: $(\lambda y, z)D(f(x, B), g, g)$

Q: $(\lambda y, z)D(f(x, C), f, G(y, y(H_1(z)), \ldots y(H_n(z)))$

Obviously the substitution $f \rightarrow g$ (or $g \rightarrow f$) is required, and this leaves us with essentially the same WFTs as in Example 12.

Section 2. A Successful Method

Suppose we are to match P and Q, and among the WFPs of P and Q which must be matched to each other we have $f(S_{i,1}, \ldots S_{i,n})$ and $f'(T_{i,1}, \ldots T_{i,n})$ for $i = 1, 2, \ldots k$. Arrange the arguments so that for every $i, S_{i,j}$ and $T_{i,j}$ are eligible "hiding" places (see p.) for $1 \leq j \leq m$, and $S_{i,j}$ and $T_{i,j}$ match for $m < j \leq p$, m and p being the largest integers for which this is possible.

Now we temporarily ignore these parts of P and Q and match the rest of P and Q. Each match may require a substitution $f \rightarrow (\lambda u_1, \ldots u_n)R(u_1, \ldots u_n)$. If in R there is an occurrence of a u_k, $p < k \leq n$, which is not inside the scope of any u_j, $1 \leq j \leq m$, then that substitution must be rejected as incompatible with the matching of $f(S_{i,1}, \ldots S_{i,n})$ and $f(T_{i,1}, \ldots T_{i,n})$. If u_k, $1 \leq k \leq p$, or $u_k(Y_1, \ldots Y_q)$, occurs not in the scope of any u_j, $1 \leq j \leq m$, then we must reconcile the matching of $S_{i,k}$ and $T_{i,k}$ (or $S_{i,k}(Y_1, \ldots Y_q)$ and $T_{i,k}(Y_1, \ldots Y_q)$) with the remaining substitutions in that particular match of P and Q. If u_k, $p < k \leq n$, or some u_h, $m < h \leq p$, for which the matching of $S_{i,h}$ and $T_{i,h}$ is incompatible with the matching of the rest of

P and Q, occurs within the scope of u_j, $1 \leq j \leq m$, then we must reconcile the elimination of that term with the rest of the substitutions.

Any potential match which is not thrown out on one of the above grounds can be completed. The tentative substitution for f must have the form $f \rightarrow (\lambda u_1, \ldots u_n)Z(u_1, \ldots u_p, u_1(Z_{1,1}(u_1, \ldots u_n)), \ldots u_1(Z_{1,n_1}(u_1, \ldots u_n)), \ldots u_m(Z_{m,1}(u_1, \ldots u_n)), \ldots u_m(Z_{m,n_m}(u_1, \ldots u_n)))$, where some of the exhibited WFTs may be absent, but none occur inside each other. We can match P and Q by substituting $f \rightarrow (\lambda u_1, \ldots u_n)h(u_1, \ldots u_p, u_1(f_{1,1}(u_1, \ldots u_n)), \ldots u_m(f_{m,n_m}(u_1, \ldots u_n)))$, followed by substituting $h \rightarrow (\lambda u_1, \ldots u_b)Z(u_1, \ldots u_b)$, $f_{i,j} \rightarrow (\lambda u_1, \ldots u_n)Z_{i,j}(u_1, \ldots u_n)$ and also making those substitutions which match S_i and T_i, $1 \leq i \leq p$, and those which eliminate the arguments of S_j, $1 \leq j \leq m$, which must be eliminated.

Thus, in order to tell how long a substitution we need to match all the pairs $f(S_i,1, \ldots S_i,n)$ and $f(T_i,1, \ldots T_i,n)$, we must examine the substitutions required for f by the rest of P and Q, and count the number of times an argument occurs with other arguments inside it.

BIBLIOGRAPHY

[1] Church, Alonzo, The Calculi of Lambda-Conversion, Annals of Mathematics studies, No. 6, Princeton University Press, Princeton, New Jersey, 1941.

[2] Bennett, J. H., Easton, W. B., Guard, J. R., Loveman, D. B., and Mott, T. H., "Semi-Automated Mathematics: SAM IV", Scientific Report No. 3, AFCRL 64-827, October 15, 1964.

[3] Guard, J. R., "Automated Logic for Semi-Automated Mathematics" Scientific Report No. 1, AFCRL 64-411, March 30, 1964

[4] "CRT-Aided Semi-Automated Mathematics", Semi-annual Report, Applied Logic Corporation, December 31, 1965.

BLANK PAGE

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Applied Logic Corporation, One Palmer Sq.
Princeton, New Jersey

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE

A Matching Procedure for W-Order Logic

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Interim Scientific Report

5. AUTHOR(S) *(Last name, first name, initial)*

Gould, William Eben

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
15 October 1966	35	4

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
ARPA Order 710 AF19(628)-3250 b. PROJECT ABXXXXXXX 8672 c. DOD ELEMENT 6154501R d. DOD SUBELEMENT N/A	Scientific Report #1 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* AFCRL-66-781

10. AVAILABILITY/LIMITATION NOTICES

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
prepared for Hq. AFCRL, OAR (CRB) United States Air Force L. G. Hanscom Field, Bedford, Mass	Advanced Research Projects Agency

13. ABSTRACT

Formation rules are presented for an W-order logic with λ operator in which each term has a well defined type. Transformation rules are given defining λ-conversion, special conversion, substitution, and alphabetic change of bound variables. The notion of one term being an "instance" of another is then defined using these transformation rules. The main problem of the paper is to develop a "matching procedure" for pairs of terms, so that every common instance of the two given terms is an instance of some term produced by the matching procedure. This is accomplished by analysing the terms according to the formation rules, matching the outermost parts first, then proceeding inductively inward. Examples are given to show that an infinite set of terms may be needed for this purpose. Conditions under which such an infinite set is needed are discussed. A procedure is developed for producing at least one common instance, if one exists.

DD FORM 1 JAN 64 1473

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Symbolic matching by computer. Omega-order logic, matching in _____ Man-machine mathematics, techniques for _____						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U).*

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.